International Journal of Production Research
Vol. 00, No. 00, 00 Month 2013, 1–23

RESEARCH ARTICLE

Construction heuristics for generating tool paths for laser cutters

Reginald Dewil*, Pieter Vansteenwegen and Dirk Cattrysse
KU Leuven, Centre for Industrial Management/Traffic & Infrastructure, Celestijnenlaan 300a, 3001 Leuven, Belgium
(*Received 00 Month 200x; final version received 00 Month 200x*)

This paper deals with generating paths for cutting irregular parts nested on thin or thick metal sheets. The objective is to minimize the total time required to cut all parts from the metal sheet explicitly taking the cost of piercing and pre-cutting into account. The problem is modelled as a generalized travelling salesperson problem with special precedence constraints. A set of construction heuristics is presented that incorporates the constraints originating from inner-outer contours, common cuts, piercing points and pre-cuts. Computational tests on a set of real-life cutting problems show that our solution approach is able to generate tool paths that for thick plates spend on average 33.4% less time than those generated by a commercial package for air movements, pre-cuts and sharp angle macros with cutting and piercing times being equal.

Keywords: laser cutting, cutting path,tool path, heuristics, precedence constraints

1. Introduction

Laser cutting is one of the major cutting processes used to manufacture sheet metal products. A typical production process consists of designing parts, nesting the parts on metal sheets, cutting the parts from the sheets and if there are three dimensional parts, bending the parts. Verlinden *et al.* (2007) describe a set of integrated production scheduling methods to minimize the total cost incurred for the entire production process. One of the results is the assignment of parts to a metal sheet after which CAM software executes the actual nesting and generates a cutting path. In sheet metal laser cutting, a typical cutting process can take between several minutes to several hours, depending on the number of parts on the plate, the material type, the machine, the process parameters, and the plate thickness. Previous literature on tool paths for cutting machines mainly deals with the so called *continuous cutting problem*(CCP) or with the generalized travelling salesperson problem (GTSP: Srivastava *et al.* (1969)).

*Corresponding author. Email: reginald.dewil@cib.kuleuven.be

ISSN: 1478-6451 print/ISSN 1478-646X online

DOI: 10.1080/1478645YYxxxxxxx
http://www.informaworld.com

In the GTSP approach, the laser head can initiate a contour at a pre-defined set of points, but once a contour is initiated, it needs to be cut completely before moving on to another contour. This approach is taken by Castelino *et al.* (2003), Kim *et al.* (2004), Wang and Xie (2005), Rao *et al.* (2006), Xie *et al.* (2009), Yang *et al.* (2010), and Jing and Zhige (2013). In the CCP the laser head can initiate cutting at any location and like in the GTSP approach, once a contour is initiated, it needs to be cut completely before moving on to another contour. This is approach is taken by Hoeft and Palekar (1997), Kumar (1998), Dror (1999), Lee and Kwon (2006), and Han and Na (1999).

In this paper, the *endpoint cutting problem* (ECP) is considered. In the ECP, the cutting tool can enter and exit the contour only at some predefined points but it may cut the contour in sections, or in other words, pre-empting is allowed. Obviously, allowing pre-empting significantly increases the number of possible solutions and makes the tool path optimization process more difficult.Wire-EDM (electric discharge machining) is a cutting process where the cutting tool never stops cutting and pre-emption is needed in order to generate feasible tool paths. Moreira *et al.* (2007), Imahori *et al.* (2008), and Rodrigues and Soeiro (2012) model the problem as a Rural Postman Problem. This solution approach, however, cannot readily be applied to thick plate laser cutting as considerable differences in technical constraints remain, such as the existence of piercings, pre-cuts, inner-outer contour relations and sharp angle costs. Similar problems are also considered by Hatwig *et al.* (2012) for combined laser cutting and laser welding and by Erdös *et al.* (2013) for laser welding. The approaches developed in these applications also do not take pre-cuts, inner-outer contour relations and sharp angle costs into account.

In Dewil *et al.* (2011), an IP model and several heuristics were proposed to generate tool paths that minimize the total distance travelled. This is a valid approach for thin plates[1] where an initial cut in the plate can be made *on the fly*, i.e. without piercing. In thicker plates however, the laser head needs to come to a complete stop and a piercing needs to be made which entails a considerable cost. This paper will consider the endpoint cutting problem with piercing costs as well as additional practical precedence constraints.

Other applications that do not utilize pre-cuts or contains precedence constraints,

2. Problem analysis

The objective of the laser cutting tool path problem is to determine a tool path that minimizes the total time required to cut all parts. A part consists of an outer contour and possibly a set of inner contours. Each contour itself is composed of a number of elements: lines or arcs. For the tool path problem, elements are defined by its start and end nodes and can be cut in both directions. An element and cutting direction combination is further referred to as a *cut* or as an *element-direction*. This total time consists of the total time required to execute all cut moves, air moves, piercings, pre-cuts and sharp angle macros:

- **cut move**: a movement where the laser head is actually cutting. The cutting time can be considered to be independent of the chosen tool path.
- **air move**: a movement where the laser head is not cutting. The time required for an air move is considerably less than the time required for a cut move. Because of

[1]The exact definition of thin, thick or very thick plates depends on the material and laser cutting machine combination.

acceleration and deceleration effects the time required for an air move is approximated by a non linear function.

- **piercing**: every time the laser head needs to start cutting in a new section of the sheet a piercing needs to be made with the exception of very thin plates. In this paper only thicker plates are considered.
- **pre-cut**: a pre-cut can be used to avoid a piercing if the possible piercing location has already been visited by the laser head earlier in the cutting process. A pre-cut is a small cut that the laser head makes in the element that later on has to be cut. Such a pre-cut requires considerably less time than a piercing.
- **sharp angle macro**: in very thick plates, a special macro is executed if a sharp corner has to be cut. In a sharp corner, the laser head slows down, until it reaches the corner point, executes a pulsed piercing, cuts a very small distance at a lower speed and intensity before regaining full cut speed.

Besides minimizing the above costs, the tool path problem is subject to precedence constraints coming from the fact that once a contour is completely cut, it detaches from the rest of the plate. This detached area can possibly shift position and hence becomes inaccessible for further cuts. A few contours on the plate however are held in place by clamps and as such are not subject to these precedence constraints.

The first set of precedence constraints comes from simple inner-outer contour relations that, as depicted in figure 1, can come from holes in parts (a), parts nested in holes (b) and parts nested in islands (c). Islands are waste areas that have become completely enclosed by other parts because of common cut nesting. Common cuts are the result of efficient nesting algorithms that place parts so close to one another that they share elements. This has the benefit that only one cut has to be made instead of two cuts.

Such common cuts are also the origin of the second set of precedence constraints. As can be seen in figure 1(c), each common cut (colored red) is enclosed by a contour composed of the common cut's two contours. The constraint is that a common cut needs to be cut before its composite contour (colored blue) is completely cut.[1] Islands that are formed through common cut nesting are also considered to be contours and it follows then that all of an island's elements have to be considered as common cuts (purple).

Common cuts also require that extra pre-cuts need to be placed. As depicted in figure 2, if a contour is completely cut (a,b,c), it can shift into the cut kerf. In that case, adjacent elements can not be cut exactly up to the cut contour because the detached contour might have shifted into the cut kerf. In order to then cut the adjacent element, the laser (because it is a cylindrical beam) would need to move into the cut kerf (d) possibly damaging the part if it shifted into the cut kerf. To avoid having to move into the cut kerf, pre-cuts are made in the adjacent elements when cutting the contour. The addition of this technical requirement, significantly increases the complexity of the tool path problem. As a consequence, the cost of cutting a common cut element depends on whether the adjacent elements are cut before or after one of the common cut's contours is completely cut.

[1]If multiple contours are nested in common cut with one another, composite contours consisting of more than two contours can be identified and also here exists the constraint that all common cuts within such a composite contour have to be cut before the last element of the composite contour is cut. This quickly leads to a very large number of constraints. However, if none of the common cuts within such a composite contour is the last element of both its contours to be cut, then the composite contour will not be closed before the common cuts within are all cut. Since, if it would contain uncut common cuts, it would imply that when the last common cut within this area is cut, that cut would close both its contours, violating the initial assumption that no element closes both its contours.

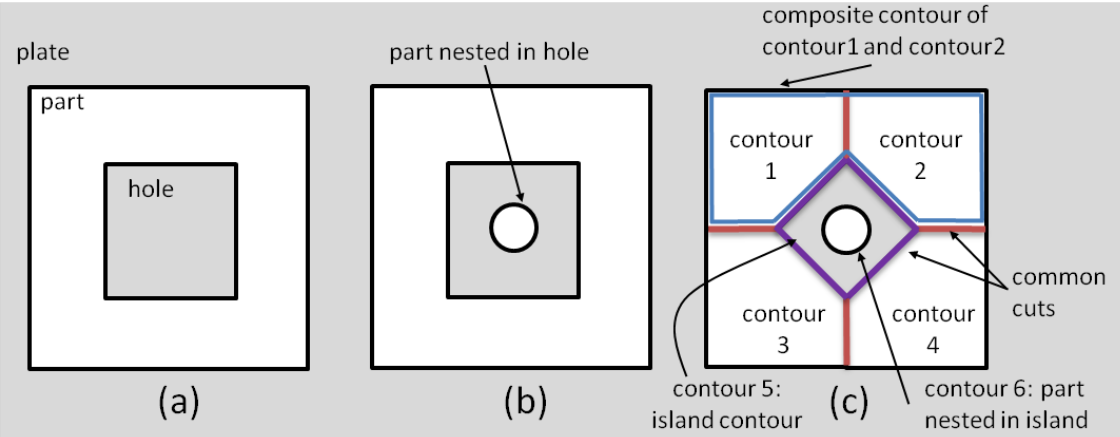

Figure 1. Parts, holes and common cuts that lead to precedence constraints

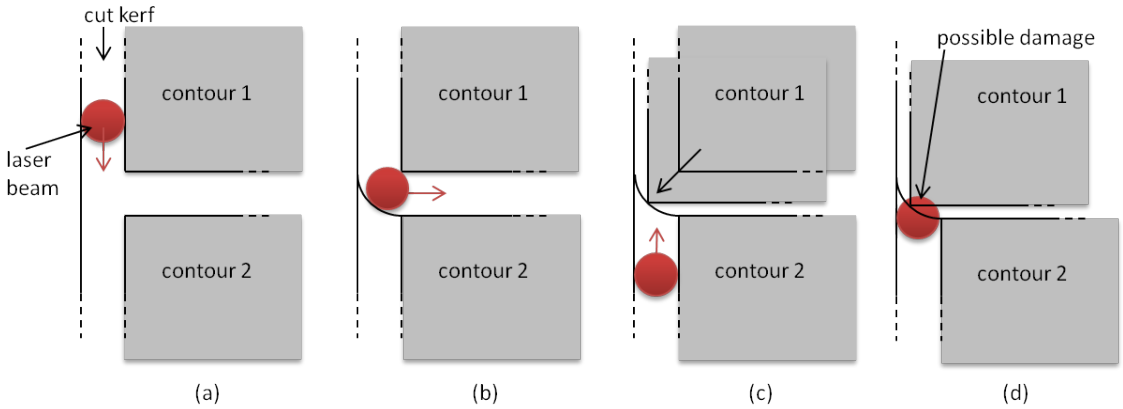

Figure 2. Shifting of parts into the cut kerf can lead to damage if the part is nested in common cut with another part

3. GTSP IP model

The IP model proposed by Dewil *et al.* (2011) can be extended to include piercing and pre-cut costs while preserving the precedence constrained generalized travelling salesperson structure (PCGTSP). In the GTSP, cities are clustered in districts and the objective is to find the shortest path that visits exactly one city of each district. The precedence constrained version is subject to precedence constraints in between districts, in between cities and between districts and cities.

3.1. *Modelling the technical specifications of laser cutting*

Figure 3 depicts how each element can be considered as a district of two cities where each city represents a cutting direction of the element. Such a city is sometimes also referred to as a *cut* or *element-direction*.

Figure 4 shows how inner-outer and common cut precedence constraints can be enforced. An inner-outer contour constraint is the requirement that each of an inner contour's districts (elements) needs to precede the finalization of its outer contour. Since a contour is not entirely cut until the last element of that contour is cut and it is unknown

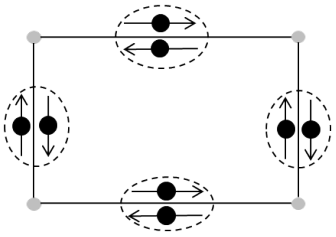

Figure 3. GTSP representation contour

in advance which element of the contour will be cut last, a *contour city* is introduced. Such a *contour city* has to be visited immediately after the last element of a contour is cut. By consequence, the inner-outer contour constraint requires that each element of the inner contour needs to be visited before the *contour city* is visited. In order to force visiting the *contour city* directly after the last element of the outer contour, each element of the outer contour has to precede the *contour city* and the *contour city* can only be entered through one of the original districts of the outer contour. Additionally in order to ensure that the correct distance is used when travelling to any other city after the *contour city* is visited, a dummy counterpart is added for each district/city of the outer contour. The dummy cities can only be entered from their counterparts or from the *contour city*. Likewise, each original city can only be exited to its dummy counterpart or to the *contour city*. An extra set of districts (depicted in purple in 4) are then added to ensure that the contour city exits to the dummy counterpart of the city that entered it. Adding these districts results in cities being present in more than one district. This however, does not invalidate the GTSP structure as Drexl (2007) shows that a GTSP problem with overlapping clusters can be transformed into a problem with disjoint clusters. Modelling inner-outer contour precedence constraints this way also ensures that a common cut can not simultaneously close both its contours, since it is impossible to visit both its contours immediately after visiting the common cut element.

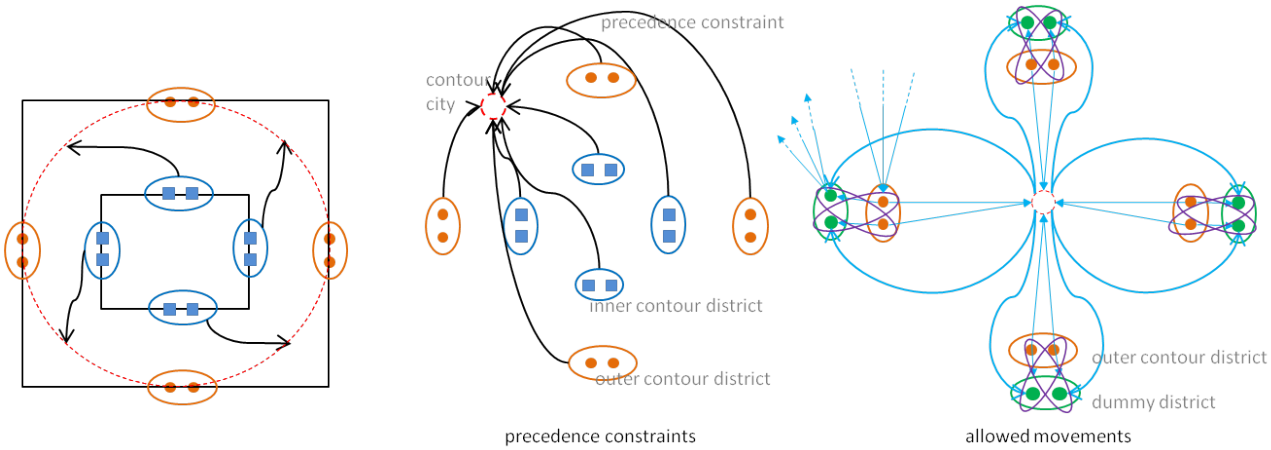

Figure 4. Modelling inner-outer contour constraints

In order to facilitate the discussion on how piercing costs, pre-cut costs and the second set of common cut precedence constraints can be added, the dummy counterpart cities of elements are temporarily ignored. Consider figure 5 which consists of six contours (a,b,c,d,e,f) that are placed "in common cut" with one another i.e. the contours share elements. Extra districts are added before and after each element district representing

the entry/exit method of the element. An entry/exit district consists of four cities representing a pierce entry, a normal entry, a pre-cut entry and a normal exit. The cut cities can only be entered or exited through one of the cities from the entry/exit districts.

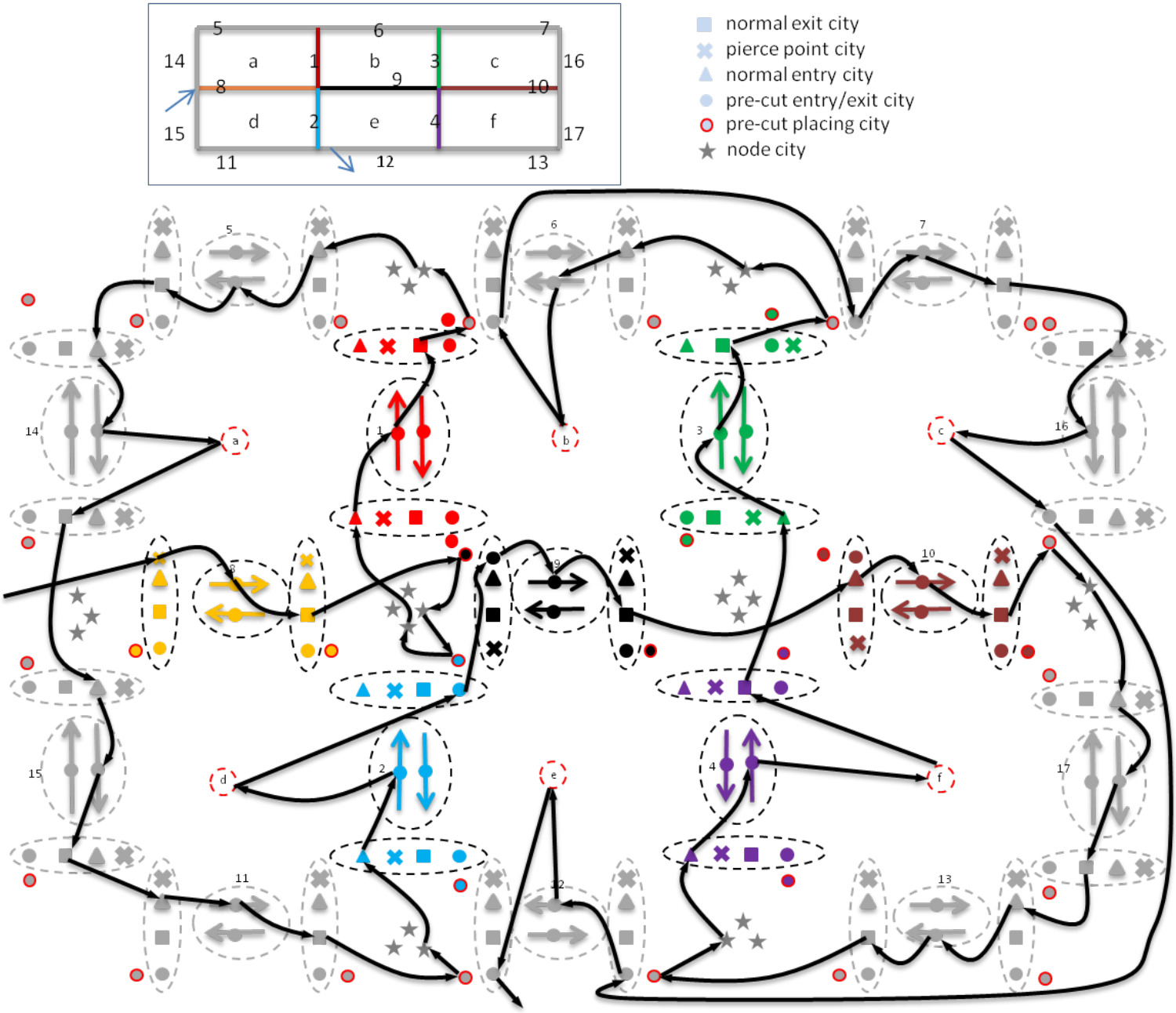

Figure 5. Modelling entry/exit constraints & costs, the black arrows indicate a feasible path

If a pre-cut is used to enter or exit an element-direction, the pre-cut has to be placed prior to this. Therefore, a *pre-cut placing city* is introduced for each possible pre-cut (and a corresponding dummy city for when the pre-cut is not visited). A precedence constraint is added ensuring that a *pre-cut city* can only be visited if its *pre-cut placing city* has already been visited.

Since an intersection can only be visited if none of the adjacent contours have been cut, a *normal exit city* can only be visited if it precedes the adjacent *contour cities*. For example, consider the *normal exit city* of element 8 at the intersection point of contours a, b, d and e. This *normal exit city* needs to precede *contour cities* b and e because if the element 5 is exited normally, the laser will partially enter the cut kerf of elements 1,2 and 9. If contours b and e are already cut completely, it might damage the part if it shifted into this corner point.

For the same reason, a *pre-cut placing city* needs to be entered from a *normal exit city* or from a *node city*. *Node cities* represent the possibility, that after a pre-cut is made, another pre-cut can be made at the same intersection. But since this requires going back to the intersection, it can only be done if none of the intersection's adjacent *contour cities* have been cut. Therefore, precedence constraints are introduced that disallow visiting a *node city* after one of the intersection's adjacent contours has been completely cut. Furthermore, a *node city* can only be entered from an adjacent *pre-cut placing city*.

Moreover since, every *node city* does not have to be visited in every solution, dummy cities are introduced that represent "not visiting the node cities" to maintain the GTSP structure. These dummy cities are not depicted in figure 5 to maintain clarity.

As an example, consider in figure 5 the following element path: [8,1,5,14,15,11,2,9,10,17,13,4,3,6,7,16,12] which corresponds to the path represented by the black arrows. The laser head first enters a *piercing city* at an entry/exit district of element 8, then it visits one of the cut cities of element 8, followed by a *normal exit city* of element 8. The laser head then visits a *pre-cut placing city* of element 9, since when element 9 needs to be cut, contour *a* will have been cut and in order to avoid entering contour *a*'s cut kerf, a *pre-cut entry city* of element 9 will have to be used. Similarly, a *pre-cut exit* will have to be used to exit element 2 later on. Hence, a *pre-cut placing city* of element 2 needs to be visited and since a *pre-cut placing city* of element 9 has just been visited, the only way to enter the *pre-cut placing city* of element 2 is by going through a *node city* at the intersection (which ensures that no adjacent contours have already been cut, thus also preventing the entering of a cut kerf after placing a pre-cut). After visiting the *pre-cut placing city* of element 2, the laser head needs to go through the intersection again, visiting another *node city*, to move to a *normal entry city* of element 1. The laser head then visits the cut city of element 1, a *normal exit city* of element 1, a *pre-cut placing city* of element 6 and a *node city* at that intersection. It then visits the *normal entry city*, cut city and *normal exit city* of element 5 after which it enters the *normal entry city* and cut city of element 14 and because element 14 is the last element of contour *a*, the *contour city* of *a* is visited next. The rest of the path is analogous to this first part.

3.2. *Mathematical model*

This model of the tool path problem can be written as an integer program where following decision variables are used:

- x_{ij} equals 1 if a move is selected from city i to city j, 0 otherwise.
- y_{IJ} equals 1 if district/element I precedes (not necessarily immediately) district/element J, 0 otherwise
- $\overleftarrow{i}$ and $\overrightarrow{i}$ represent the two cutting directions of district/element I
- $\acute{I}$ and $\grave{I}$ represent the two entry/exit districts of element I
- $\underline{i}$ is the dummy equivalent of city i
- $\dot{p}$ represents the pre-cut placing city of pre-cut p

The objective function minimizes the sum of the total travel time where the distance matrix has been modified to include the piercing, pre-cut placing and sharp angle costs when entering the respective cities.

$$Min \sum_i \sum_j d_{ij} x_{ij} \tag{1}$$

$$\sum_j x_{\overleftarrow{i}\,j} + x_{\overrightarrow{i}\,j} = 1 \qquad \forall \text{ elements I} \tag{2}$$

$$\sum_j x_{j\,\overleftarrow{i}} + x_{j\,\overrightarrow{i}} = 1 \qquad \forall \text{ elements I} \tag{3}$$

$$\sum_i x_{ij} = \sum_k x_{jk} \qquad \forall \text{ cuts j} \tag{4}$$

$$y_{IJ} \geq x_{\overleftarrow{i}\,\overrightarrow{j}} + x_{\overleftarrow{i}\,\overleftarrow{j}} + x_{\overrightarrow{i}\,\overrightarrow{j}} + x_{\overrightarrow{i}\,\overleftarrow{j}} \qquad \forall \text{ elements I,J} = 2...\text{n; I} \neq \text{J} \tag{5}$$

$$y_{IJ} + y_{JI} = 1 \qquad \forall \text{ elements I,J} = 2...\text{n; I} \neq \text{J} \tag{6}$$

$$y_{IJ} + y_{JK} + y_{KI} \leq 2 \qquad \forall \text{ elements I,J,K} = 2...\text{n; I} \neq \text{J} \neq \text{K} \tag{7}$$

$$y_{1I} = 1 \qquad \forall \text{ elements I} = 2...\text{n} \tag{8}$$

$$x_{ij} \in \{0,1\} \qquad \forall \text{ cities i,j} \tag{9}$$

$$y_{IJ} \in \{0,1\} \qquad \forall \text{ districts I,J} \tag{10}$$

$$y_{Jc} = 1 \qquad \forall \text{ contour cities } c, \text{ J} \in c \tag{11}$$

$$\sum_{\underline{j} \in c} x_{c\underline{j}} = 1 \qquad \forall \text{ contour cities c} \tag{12}$$

$$\sum_{i \in c} x_{ic} = 1 \qquad \forall \text{ contour cities c} \tag{13}$$

$$x_{c\,\overrightarrow{\underline{i}}} + x_{\overrightarrow{i}\,\overrightarrow{\underline{i}}} = \sum_k x_{k\,\overrightarrow{i}} \qquad \forall \text{ cuts i} \tag{14}$$

$$y_{kc} = 1 \qquad \forall \text{ k} \in \text{inner contour of c} \tag{15}$$

$$\sum_{l \in \acute{I}} x_{l\,\overrightarrow{i}} = \sum_k x_{k\,\overrightarrow{i}} = \sum_{r \in \grave{I}} x_{\overrightarrow{\underline{i}}\,r} \qquad \forall \text{ elements I} \tag{16}$$

$$\sum_k x_{kp} = \sum_k x_{k\dot{p}} \qquad \forall \text{ pre-cut cities p,}\dot{p} \qquad (17)$$

$$\sum_k x_{k\dot{p}} = y_{\dot{p}p} \qquad \forall \text{ pre-cut cities p,}\dot{p} \qquad (18)$$

$$\sum_k x_{k\dot{p}} + \sum_k x_{k\underline{\dot{p}}} = 1 \qquad \forall \text{ pre-cuts p} \qquad (19)$$

$$\sum_k x_{k\underline{\dot{p}}} = y_{end,\dot{p}} \qquad \forall \text{ pre-cut placing cities } \dot{p} \qquad (20)$$

$$\sum_k x_{k\dot{p}} = \sum_{n_{norm.exit} \in N} x_{n\dot{p}} + \sum_{t_{nodecity} \in N} x_{t\dot{p}} \qquad \forall \text{ intersections N, } \dot{p} \in \text{N} \qquad (21)$$

$$\sum_k x_{kn} = \sum_{p \in N} x_{\dot{p}n} \qquad \forall \text{ node cities n} \qquad (22)$$

$$\sum_k x_{kn_{nodecity}} = y_{n_{nodecity}c} \qquad \forall n_{nodecity} \in \text{N, C adjacent to N, } \forall \text{ N} \qquad (23)$$

$$\sum_k x_{k,\text{normal exit n}} = y_{ic} \qquad \forall \text{ n, i} \prec \text{ n, C adjacent to n} \qquad (24)$$

$$\sum_k x_{kn} + \sum_k x_{k\underline{n}} = 1 \qquad \forall \text{ node cities n} \qquad (25)$$

$$\sum_k x_{kn} = 1 - y_{end,N} \qquad \forall \text{ node cities n} \qquad (26)$$

Constraints:

Constraints (2)-(10) are based on a GTSP formulation of Sherali *et al.* (2005) that readily allows the modelling of precedence constraints. (2) and (3) respectively ensure that every element is exited and entered exactly once.

(4) states that if an element-direction is entered it has to be exited.

(5),(6),(7),(8) are sub tour elimination constraints.

(9) constrains the x variables to a binary value.

(10) constrains the y variables to binary values.

(11) ensures that every element precedes the contour city of the contour it is part of.

(12) ensures that every contour city exits to one of its cut's dummy counterparts.

(13) ensures that every contour city is entered by one of its original cut cities.

(14) ensures that if a cut is entered from any other city k, its dummy counterpart city will be entered by either the contour city or the original cut city.

(15) ensures that all elements of a contour precede its contour's outer contour city.

(16) forces that if a cut is entered from any other city k, it is entered through one of its entry cities and that its dummy counterpart exits to one of its exit cities.

(17) ensures that if a pre-cut city is entered from any other city k, its pre-cut placing city is also entered.

(18) ensures that if a pre-cut city is entered from any other city k, the pre-cut placing city precedes the pre-cut city.

(19) and (20) are added to maintain the GTSP structure. If a pre-cut placing city isn't visited, its dummy counterpart needs to be visited after the actual tool path.

(21) ensures that a pre-cut placing city can only be entered either from a normal city or

a node city at the pre-cut placing city's intersection.
(22) ensures that a node city can only be entered from an adjacent pre-cut placing city.
(23) models the precedence constraint that if a node city is n at intersection N is visited by any other city k, it needs to precede all contour cities adjacent to intersection N.
(24) if a normal exit city is visited, its element district needs to precede the adjacent contour cities.
(25) and (26) are in order to maintain the GTSP structure. If a node city isn't visited, its dummy counterpart needs to be visited after the actual tool path.

Implementations of this program in CPLEX confirmed our expectations that the calculation times were too large to be of any practical use on the production floor. Therefore, a heuristic approach was chosen.

4. Generating tool paths

In a first approach, the construction and improvement heuristics presented in Dewil *et al.* (2011) were adapted to include piercing and pre-cut costs. Initial tests showed that a prohibitive amount of computation time was spent on evaluating local moves. Since the total computation time is important for practical implementations, only a limited amount of local moves can be executed and therefore the quality of the final solution becomes heavily dependent on the quality of the initial solution. For this reason, four heuristic sub procedures were developed that depending on how these are combined, can generate different tool path patterns. In order to facilitate the discussion of the heuristics, a *pierce group* is defined as an entity that requires at least one piercing[1]. This can be a single contour or a group of contours that are connected to each other by common cuts as depicted in figure 6.

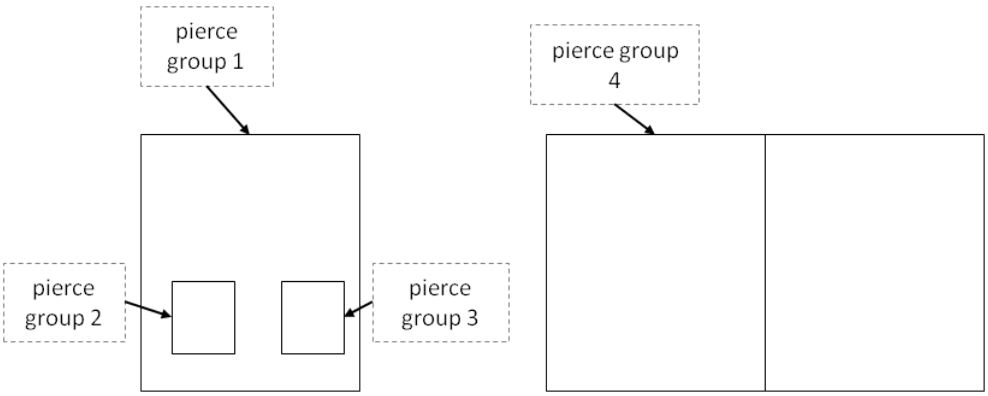

Figure 6. Pierce groups

Sub procedure 1: given a set of pierce groups and start and end positions, this sub procedure generates a GTSP solution. It employs the cheapest insertion heuristic which was adapted for GTSP application by Fischetti *et al.* (1995) from the cheapest insertion heuristic for the TSP as analyzed by Rosenkrantz *et al.* (1974). The cheapest insertion heuristic generates an initial tour with chosen entry and exit points (which can

[1]Since piercings are so costly, the optimal solution of most practical instances will have exactly one piercing per pierce group.

be different from one another in the case of common cut pierce groups). The GTSP tour construction is followed by a 2-opt local search phase similar to the G2-opt heuristic of Renaud and Boctor (1998).

Sub procedure 2: given a set of pierce groups and start and end positions, this sub procedure iteratively inserts one of the pierce groups of the set between two cuts in the partial path. This sub procedure is similar to sub procedure 1 in the sense that it is also an insertion procedure. But the difference lies in the fact that the insertion occurs between individual cuts and not between pierce groups. In every iteration the sub procedure evaluates the insertion of every remaining pierce group for every possible entry and exit node and for all positions between the start and end positions. When a pierce group, entry node, exit node and insertion position is selected, a sub path is determined for the pierce group. This sub path is then inserted at the chosen position and the process continues until all pierce groups have been inserted.

Sub procedures 3 and 4 both generate sub paths for common cut pierce groups.

Sub procedure 3: also referred to as the *Part-by-Part* sub tool path heuristic: given an entry and exit point for a pierce group containing common cuts, the objective of this sub procedure is to find a short tool path that does not violate any precedence constraints. Furthermore, since it is modelled after the sub tool path heuristics of CAD/CAM software packages, it attempts to cut a contour as quickly as possible once it is initiated. Such a sub tool path can be determined by iteratively selecting a contour that is adjacent to the current location of the laser head and if cut, does not create an infeasibility. If there is no adjacent contour left uncut or it would create an infeasibility if cut, a contour is selected from a list of eligible contours. The contour that is selected, is the one that contains an already visited node that is closest to the laser head's current position. If there would be a tie between several contours, the contour that is furthest away from the pierce group's exit node is selected. The selected contour is then added to the sub tool path, the laser head's current position is updated and the list of eligible contours is expanded by all contours that are adjacent to the recently added contour and are not yet present in the list or have already been cut. As an example, consider figure 7 where node

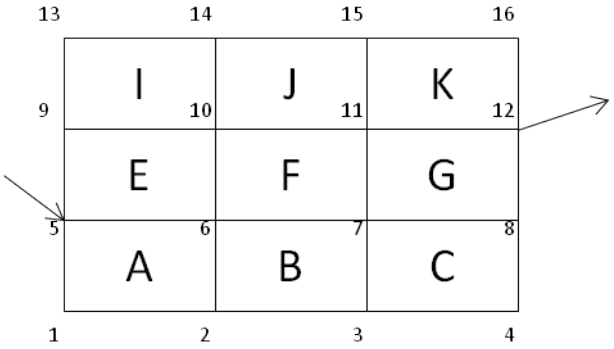

Figure 7. Sub tool path determination without pre-empting

5 and node 12 are determined to respectively be the entry and exit nodes. The eligible contour list is then initialised with contours A and E. Suppose contour E is selected to be cut and since there is no difference with regards to feasibility whether the contour is cut clockwise or counter clock wise, a direction is chosen randomly. This results in the laser head being positioned again in node 5. The list of eligible contours is updated by adding contours I,J,F and B. Since the laser head is positioned in node 5 and cutting contour A would cause no infeasibility, contour A is selected to be cut which results in the laser

head ending up in node 6. No change in the list of eligible contours occurs. In node 6, both contours F and B can be feasibly cut which would result in the laser head ending up in nodes 10 or 2 respectively. Because of the non linear nature of air moves, both nodes 10 and 2 are equally far from the exit node and therefore one is randomly chosen. Suppose F is selected. The laser head then ends up in node 10 and contours K,G and C are added to the list of eligible contours. Next, contour I is cut and the laser head ends in node 9. From here, the closest air move that doesn't cause an infeasibility is towards node 14 where contour J is cut, positioning the laser head in node 11. This process is repeated until all contours are cut after which any necessary pre-cuts are added to the sub tool path. This sub procedure always yields feasible tool paths but the tool paths will not always exit the pierce group at the chosen exit node. The exit node functions more as a guide to exit the pierce group in its vicinity. In practice, one could effectively also use the entry node of the next pierce group in the pierce group order. Tests did not show a difference in sub tool path quality.

Sub procedure 4: given entry and exit points for a pierce group containing common cuts, the objective of this sub procedure is to find a short tool path that does not violate any precedence constraints. In this case, the sub tool path is not constrained to finish a contour as quickly as possible. The sub procedure works similarly to the Element-Insertion heuristic described in Dewil *et al.* (2011) except for the inclusion of piercing and pre-cut costs. This inclusion increases the evaluation time of inserting a cut, since the heuristic now needs to check for each possible insertion location whether the adjacent cuts have already been inserted in the partial path. The evaluation of a cut at a given position requires that the heuristic checks whether any cuts adjacent to the cut's entry node have already been inserted. If so, a check will need to be executed whether that adjacent cut can supply a pre-cut in a feasible manner. If this is valid, no piercing cost will be required. If multiple cuts adjacent to the entry node have already been inserted, multiple checks will need to be executed to see which element can supply the cheapest pre-cut taking sharp angle costs into account. Similarly, any cuts adjacent to the cut's exit node visited after the insertion position can possibly enjoy a discount if the cut can supply a cheaper pre-cut than the current entry method of the adjacent cut.

Five different heuristics were built using the above described sub procedures, each giving rise to a different pattern in the tool path. The developed heuristics, depicted in figure 8 are: NPInf, NPPbP, PInf, PPbP and PFr. The NPInf and NPPbP heuristics generate tool paths that disallow pre-emptions and that result in an *inner contours first* pattern and a *part-by-part* pattern respectively. These tool path patterns are frequently encountered in CAD/CAM software and thus can also be used to show areas of improvement for CAD/CAM software without disrupting current practices too much. PInf and PPbP on the other hand do allow pre-emption but remain constrained to the industry's current patterns. Lastly PFr does not restrict the tool path to any pattern and allows pre-emption.

To generate tool paths without pre-emption, both sub procedures 2 and 4 can not be used since they will result in the pre-emption of pierce groups. Therefore, the NPInf and NPPbP heuristics only utilize sub procedures 1 and 3.

NPInf cuts all innermost contours across the entire plate first, then it cuts all contours across the plate one level higher and so on until all contours have been cut. This requirement makes sense if heat accumulation is a problem, which is the case in very thick plates. The pattern generated by this requirement allows regions to cool down before being revisited by the laser head possibly avoiding burn off of sharp corners in overheated

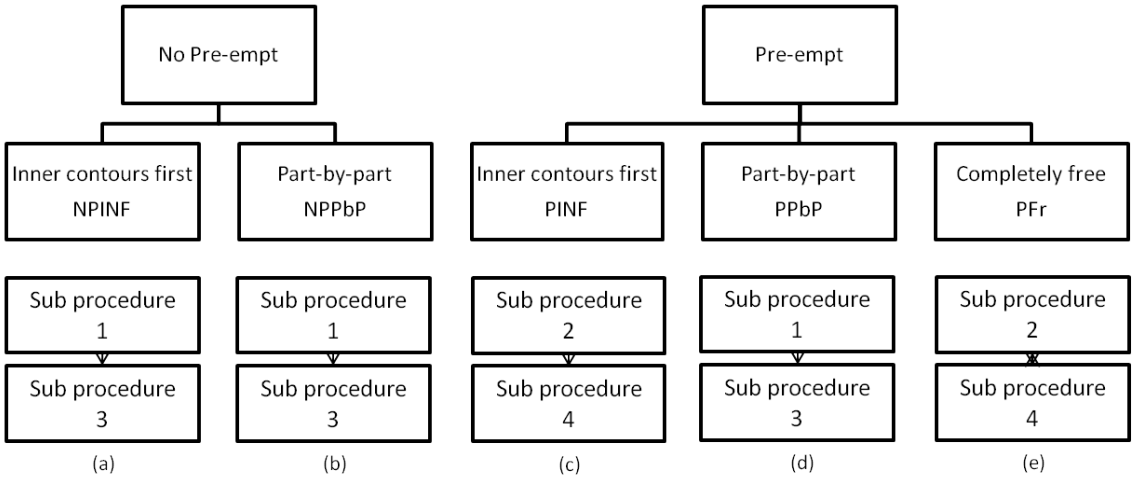

Figure 8. Construction heuristics

regions. It works by starting at the lowest level, iterating over all levels and determining a GTSP solution by using sub procedure 1 for all pierce groups of the current level. Given the entry nodes, exit nodes and pierce group order for every level, sub tool paths are determined for the pierce groups. In the case of pierce groups without common cut, this is trivial since it just entails choosing a direction to cut, but when the pierce groups contain common cuts sub procedure 3 is used.

NPPbP works according to a part-by-part strategy where once a part has been initiated, it needs to be finished. And since pre-empting is not allowed, this implies that first all inner contours of a part are cut, before moving on to a higher level until the part is completely cut. A part-by-part strategy allows for early evacuation of parts or for quality control. If an issue is identified after cutting one part, the process can then be stopped without having to discard the rest of the plate as waste since no cuts have yet been made there. The heuristic starts by generating a GTSP solution for the outermost pierce groups using sub procedure 1, followed by determining sub tool paths for this level using sub procedure 3. The heuristic then iterates backwards over the partial path built so far. If an element that is entered by a piercing is encountered, the element's pierce group's inner contours are identified, sub procedure 1 is called for these inner contours and the resulting sub tool path is inserted before the element. This continues until the start node is reached and a complete tool path is determined.

Heuristics PInf, PPbP and PFr on the other hand, do allow pre-empting.

PInf, like heuristic NPInf determines tool paths on a level-by-level basis, starting with the innermost contours but it allows pre-empting when cutting a given level. It starts similarly to the case without pre-empts by ordering all pierce groups according to their level. Starting at the lowest level, the heuristic iterates over all levels and executes a pierce group insertion procedure (sub procedure 2). If a pierce group sub tool path needs to be determined, sub procedure 4 is called.

PPbP, like heuristic NPPbP creates a tool path on a part-by-part basis, but since it allows pre-empting a part can be initiated at its outer contour. It starts by determining a GTSP solution through sub procedure 1 on the highest level pierce groups. The heuristic then iterates backwards over the partial path and if it encounters an element that closes a contour, all of the contour's inner contours are inserted also according to sub procedure 1 in a position adjacent to one of the contour's elements. If a pierce group sub tool path

needs to be determined, sub procedure 3 is called.
PFr, allows pre-empting and there are no pattern requirements. The heuristic applies a pierce group insertion procedure on the highest level pierce groups. It then iterates backwards over the partial path and if it encounters an element that closes a contour, all of the contour's inner contours are inserted according to a pierce group insertion procedure in a position anywhere before the last element of the contour. If a pierce group sub tool path needs to be determined, sub procedure 4 is called.

5. Computational results

The above described heuristics were tested on a set of 29 test instances which are depicted in appendix A (Figures A1-A29). All test instances consist of parts nested on a stainless steel plate with a thickness of 6 mm, which are considered to be thick plates.

The tool paths generated by the heuristics were also compared to tool paths generated by a commercial CAD/CAM package of which the name cannot be disclosed for confidentiality reasons. However, it is generally assumed that commercial CAD/CAM packages generate tool paths based on very simple heuristics. Therefore, in order to better evaluate the contribution of our approach versus the industry's practice of GTSP-like tool paths, the instances were transformed in TSP instances using the transformation of Noon and Bean (1991) and then solved using an improved Lin-Kernighan heuristic (Helsgaun (2000)). Helsgaun's Lin-Kernighan heuristic (LKH) has been shown to yield very high quality solutions for instances with similar problem sizes as those encountered in the tool path problem. Since the LKH does not take inner-outer contour precedence constraints into account, the generated tool paths are not feasible for all instances that contain inner contours and have to be interpreted as lower bounds for the industry's approach. For the instances containing pierce groups with common cuts, the LKH was used to determine the entry and exit nodes of the pierce groups and the Element-Insertion heuristic (sub procedure 4) was used to determine the sub tool paths of the pierce groups containing common cuts.

Tables 1, 2 and 4 summarize the results where n stands for the number of elements and *base* stands for the sum of the total cutting time of all elements and the total time required to pierce the minimum number of piercings for that instance. These times are considered independent of the chosen tool paths but are provided to give perspective. The obj(s) values are the sum of air travel times, pre-cut times and sharp angle times given in seconds of execution time. Typical solution times for the CAD/CAM solver and the heuristics on an Intel i7-2630QM 2GHz CPU range between 0.5 and 4 seconds. t_{comp} gives the computation times for the Lin Kernighan heuristic which clearly shows that for larger instances, the LKH takes up too much computation time to be of practical use on the shop floor.

The 14 instances in Table 1 are characterised by the fact that a near optimal solution for the GTSP corresponds to a near optimal solution for the end cutting problem because the parts are large relative to the inter part distances or the parts contain a large number of inner contours that are more or less evenly spread across the entire part. Additionally, these instances do not contain common cuts. From these results it can be seen that the proposed heuristics are able to generate faster tool paths than those generated by the CAD/CAM solver, thus proving a potential for improvement in the commercial solvers.

Since CAD/CAM solvers model the problem as a GTSP, it is of interest to directly compare the CAD/CAM solver to the LKH and also to the NPPbP heuristic which is

inst.	**n**	**Base**	**C/C**	**NPInf**	**NPPbP**	**PInf**	**PPbP**	**PFr**	**LKH**	
		(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	t_{comp}(s)
10b	421	524	110.0	81.4	81.4	82.4	81.4	82.4	**78.79**	6.26
12	1793	1900	517.9	298.6	301.3	303.3	301.3	300.8	**271.1**	233.08
13	289	432	105.3	94.1	90.6	94.3	102.7	96.5	**84.8**	11.32
18	656	820	179.5	147.0	146.5	146.1	153.1	146.0	**140.1**	17.46
19	1002	1513	291.7	122.9	124.1	126.5	131.6	120.4	**106.7**	81.64
20	2357	2303	1038.0	668.7	663.6	660.1	692.6	662.2	**621.8**	823.23
21	529	929	400.7	294.0	277.3	293.8	305.8	268.0	**263.6**	25.03
24	5685	2359	1427.2	793.1	781.8	798.3	791.5	775.3	**751.7**	4520.33
24b	5685	2359	1428.5	782.3	777.7	789.1	787.0	769.4	**744.0**	1239.67
25	1424	1394	449.6	298.8	291.3	300.9	307.2	295.9	**273.3**	227.82
26	2810	1871	585.9	387.9	389.3	395.0	398.5	391.8	**367.2**	917.00
27	2194	1282	486.5	311.6	306.1	311.8	319.2	308.6	**288.9**	2984.65
28	324	1282	173.6	111.0	117.0	112.0	118.9	107.4	**104.3**	17.71
30	145	1282	174.4	103.9	103.9	75.2	115.3	76.0	**70.5**	2.52
avg. gap with CAD/CAM				**34.8%**	**35.3%**	**35.6%**	**32.1%**	**36.6%**	**40.5%**	

Table 1. Computational results of the GTSP instances between the proposed heuristics, a commercial CAD/CAM software and the Lin-Kernighan heuristic

a basic GTSP heuristic. The NPPbP is able to generate tool paths that are on average 35.3% faster than the CAD/CAM solver. In comparison, the LKH generates (infeasible) tool paths that are on average 40.5% faster than the CAD/CAM tool paths, resulting in a gap of 5.2%. Of interest to note is that even in instances where one expects that a GTSP based solution will be of high quality (because of large inter part distances, small contours or the presence of many inner contours), the PFr generates tool paths that are on average 36.6% faster than the CAD/CAM tool paths. This results in a gap with the LKH of only 3.9%.

inst.	**n**	**Base**	**C/C**	**NPInf**	**NPPbP**	**PInf**	**PPbP**	**PFr**	**LKH**		**Opt.**
		(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	t_{comp}(s)	obj(s)
10	421	524	115.1	77.7	77.7	76.7	77.7	76.7	**74.1**	0.01	72.4
17	145	509	18.3	9.7	9.7	**9.3**	9.7	**9.3**	9.7	0.15	9.3
29	121	828	92.8	75.4	81.5	71.9	75.0	**66.5**	66.9	1.04	65.4
34	91	981	67.0	52.3	79.8	53.6	55.4	50.6	**49.6**	0.83	48.8
35	21	110	12.6	12.1	12.1	**11.50**	12.1	11.52	11.9	0.04	11.5
36	61	654	45.5	35.9	52.6	36.3	37.4	**33.8**	33.9	0.41	33.8
37	31	327	23.8	19.8	23.6	20.1	19.7	**17.5**	18.5	0.13	17.5
38	237	526	95.9	76.44	76.44	74.16	76.44	**74.16**	74.66	3.54	-
avg. gap with CAD/CAM				**23.2%**	**8.9%**	**24.2%**	**22.2%**	**28.0%**	**27.2%**		

Table 2. Computational results of the pre-empt instances between the proposed heuristics, a commercial CAD/CAM software and the Lin-Kernighan heuristic

The 8 instances in Table 2 are characterised by the fact that good solutions would probably contain several pre-emptions because the parts are large relative to the inter part distances. Similar to the instances in Table 1, these instances do not contain common cuts. Compared to the CAD/CAM solver, again all proposed heuristics are able to generate faster paths. The PFr heuristic generates the fastest tool paths with an average improvement of 28.0% opposed to the LKH tool paths that only results in an improvement of 27.2%. Furthermore, since the LKH does not take precedence constraints into account, the solutions generated are not necessarily feasible. These results establish a positive argument to utilize a pre-emption strategy.

Inst.	PFr	Opt	Gap
10	76.69	**72.42**	5.90%
17	**9.35**	**9.35**	0.00%
29	66.52	**65.35**	1.79%
34	50.60	**48.45**	4.44%
35	**11.52**	**11.52**	0.00%
36	33.76	**33.26**	4.64%
37	**17.52**	**17.52**	0.00%
		avg. gap:	**2.40%**

Table 3. Comparison between the PFr heuristic and the optimal solution

In order to better evaluate the efficacy of PFr heuristic, the optimal solutions for several smaller instances were determined either manually (instances 10, 29, 34, 36) or through the IP formulation (instances 17, 35, 37) and can be found in Table 3. The optimum could not be determined for the other instances because of excessive memory and computation time requirements. For these smaller instances, the PFr is able to generate solutions that have a maximal gap of 5.90% and are on average within 2.40% of the optimum. Furthermore, the PFr is able to reach the optimal solution in three of the seven instances.

inst.	**n**	**Base**	**C/C**	**NPInf**	**NPPbP**	**PInf**	**PPbP**	**PFr**	**LKH**	
		(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	obj(s)	t_{comp}(s)
11	745	574	224.8	183.8	183.7	180.4	194.0	176.1	**172.2**	36.49
14	230	314	75.9	63.7	75.3	49.2	74.3	**44.1**	53.1	4.56
15	253	360	81.5	78.6	76.7	67.8	85.0	**63.0**	68.0	2.98
16	303	371	132.3	96.3	87.9	84.0	116.0	**67.2**	72.1	2.45
16b	123	182	63.2	67	63.2	**52.0**	63.2	**52.0**	52.0	0.02
22	385	770	300.6	227.9	227.9	**191.7**	227.9	**191.7**	196.0	0.04
23	841	970	345.3	282.4	277.7	259.2	284.2	**251.2**	263.0	0.32
avg. gap with CAD/CAM			**15.3%**		**14.6%**	**26.7%**	**9.4%**	**30.9%**	**27.4**%	

Table 4. Computational results of the common cut instances between the proposed heuristics, a commercial CAD/CAM software and the Lin-Kernighan heuristic

The 7 instances in Table 4 contain common cuts. Also here, the proposed heuristics generate faster tool paths than the commercial CAD/CAM solver and the best results are obtained by the PFr heuristic with an average improvement of 30.9% opposed to an average improvement of 27.4% by the Lin Kernighan heuristic.

To better evaluate the impact of the sub tool path quality on the complete tool path quality an additional comparison with the common cut sub tool path heuristic of Manber and Israni (1984) can be made. The Manber & Israni (M&I), the Part-by-Part (PbP), and the Element-Insertion (El-Ins) heuristics were applied to the different common cut pierce groups present in instances 11,14,15 and 16. The results are summarized in Table 5 where the obj. columns give only the sub tool path execution time in seconds without considering movements to other pierce groups. The % impr. columns give the percentage improvement over the M&I heuristic.

We can conclude from Table 5 that the Element-Insertion heuristic generates the shortest tool paths in all instances. The average improvement over the M&I heuristic is 15.13%, but varies greatly between 1.47% and 31.08% depending on the instance. More in-depth investigation showed that the Part-by-Part heuristic, being constrained to finishing each contour as quickly as possible, introduced the most air moves, pre-cuts and sharp angle macros resulting in the longest execution time. The M&I heuristic introduced the minimum number of pre-cuts necessary together with the shortest air

	M&I	**PbP**		**El-Ins**	
	obj.	obj.	%impr.	obj	%impr.
$CCPG_{11}$	30.8	31.7	-2.95%	**29.0**	5.60%
$CCPG_{14}$	48.8	59.5	-22.04%	**33.6**	31.08%
$CCPG_{15}$	5.5	5.6	-2.77%	**5.4**	1.47%
$CCPG_{16}$	61.4	60.9	0.75%	**47.7**	22.36%
			-6.75%		**15.13%**

Table 5. Comparison of PbP and Element Insertion sub tool path heuristics with Manber&Israni's heuristic

moves. In comparison, the element-insertion heuristic introduced many more pre-cuts and longer air moves than the M&I heuristic, but the tool paths contained a lot fewer sharp angle costs which for plates with a thickness of 6 mm are not negligible. Therefore, we have to conclude that for plates with a thickness that does not entail sharp angle macros, the M&I heuristic will generate faster sub tool paths. But for plates with a thickness that does entail sharp angle costs, the element-insertion heuristic will result in faster sub tool paths.

All of the instances further show that no single heuristic dominates the other heuristics for all instances. The PFr yields on average the best results of all the proposed heuristics with an average improvement for the three groups of instances of respectively 36.6%, 28.0% and 30.9%. However, since calculation times for these heuristics are relatively short, all of the proposed heuristics can be executed simultaneously on the shop floor with the goal of selecting the fastest tool path for execution. Such a strategy would lead to an improvement for the three groups of instances of respectively 37.3%, 28.1%, and 30.9% compared to tool paths generated by the CAD/CAM software or on average for all instances an improvement of 33.4%. Compared to the LKH, this approach results in an average gap of 3.2%, -0.9%, and -3.5% and a maximum gap of 5.5%, 2.2%, and 1.7% for the three groups of instances.

For the *GTSP* and even the *Pre-empt* instances, the LKH results show that a pure GTSP approach might generate even better results than the PFr heuristic. However, the generated tool paths for all of the instances that contained inner contours (12, 13, 18, 19, 20, 21, 24, 24b, 25, 26, 27, 28, 29, 30, 34, 36, and 37) were found to be infeasible since the tool paths generated by the Lin-Kernighan heuristic do not take precedence constraints into account. Therefore, fast precedence constrained GTSP heuristics can be an interesting avenue for further research.

In the case of the common cut instances, the PFr and PInf heuristics clearly generate the best tool paths. This mainly results from the fact that these heuristics do not let the intra pierce group air moves go to waste. The air moves are utilized to cut other pierce groups if this is advantageous. This also results in far fewer air movements in the tool paths generated by the PFr and PInf heuristics than the Lin-Kernighan heuristic.

6. Conclusions

In this paper, the laser cutting tool path problem with piercing, pre-cut and sharp angle costs is investigated. An IP model that is able to model the complete problem has been developed and explained in detail. However, this model is found to be too complex to solve with a commercial solver within a short enough amount of time to be of practical use on the shop floor.

Therefore, four sub procedures were presented that were combined into five different

construction heuristics. The computational results indicate that the proposed heuristics are able to generate feasible tool paths that are of significantly better quality than those generated by a commercial CAD/CAM package in a similar amount of computation time.

Furthermore, additional comparisons are made with the improved Lin-Kernighan heuristic of Helsgaun (2000). The tool paths generated by the LKH result in the best possible tool paths that the industry's approach of utilizing GTSP paths can reach. The results show that in inefficient nests or nests with parts that contain many inner contours, a GTSP approach can result in tool paths that are on average 3.9% faster than those generated by the proposed heuristics. However, since this GTSP approach cannot deal with precedence constraints, all the generated tool paths for instances that contain inner contours are found to be infeasible. Therefore, these results should only be considered as infeasible lower bounds.

The results further show that for instances with large parts and efficient nests, two of the proposed heuristics do result in faster tool paths than those generated by both the commercial software and the Lin-Kernighan heuristic. This is even more the case for the common cut pierce groups since a pre-emption strategy utilizes the unavoidable intra pierce group air moves to visit other pierce groups and hence saves on the total number of air moves.

A further comparison between the PFr heuristic and optimal tool path solutions for a small number of instances showed that the PFr heuristic generates solutions that are on average within 2.40% of the optimum. However, the gap can reach up to 5.9% suggesting that improvement heuristics can still meaningfully improve on these tool paths.

Of the two sub tool path procedures proposed, it is shown that the Element Insertion procedure outperforms the sub tool path procedure described in Manber and Israni (1984). On the other hand, the Manber & Israni heuristic was found to outperform our proposed Part-by-Part sub tool path heuristic which was modelled after the sub tool path heuristics employed by CAD/CAM solvers. This suggests that commercial CAD/CAM software packages can also benefit by improving their common cut sub tool path heuristics.

Acknowledgement

The authors acknowledge the financial support from IWT-Vlaanderen (Instituut voor de Aanmoediging van Innovatie door Wetenschap en Technologie Vlaanderen).

References

Castelino, K., D'Souza, R., and Wright, P., 2003. Toolpath optimization for minimizing airtime during machining. *Journal of Manufacturing Systems*, 22 (3), 173–180.

Dewil, R., Vansteenwegen, P., and Cattrysse, D., 2011. Cutting Path Optimization using Tabu Search. *Key Engineering Materials*, 473, 739–748.

Drexl, M., 2007. On Some Generalized Routing Problems. Thesis (PhD). Rheinisch-Westfälischen Technischen Hochschule Aachen.

Dror, M., 1999. Polygon plate-cutting with a given order. *IIE Transactions*, 31 (3), 271–274.

Erdös, G., *et al.*, 2013. Planning of remote laser welding processes. *In*: *Forty Sixth CIRP Conference on Manufacturing Systems 2013*, Vol. 7, Jan.., 222–227.

Fischetti, M., Gonzalez, J.J.S., and Toth, P., 1995. A Branch-and-Cut Algorithm for the Symmetric Generalized Travelling Salesman Problem. *Operations Research*, 45 (3), 378–394.

Han, G.c. and Na, S.j., 1999. A Study on Torch Path Planning in Laser Cutting Processes Part 2: Cutting Path Optimization Using Simulated Annealing. *Journal of Manufacturing Systems*, 1 (1), 62–70.

Hatwig, J., Zaeh, M.F., and Reinhar, G., 2012. An Automated Path Planning System for a Robot with a Laser Scanner for Remote Laser Cutting and Welding. *In*: *Proceedings of 2012 IEEE International Conference on Mechatronics and Automation*, 1323–1328.

Helsgaun, K., 2000. An effective implementation of the LinKernighan traveling salesman heuristic. *European Journal of Operational Research*, 126 (1), 106–130.

Hoeft, J. and Palekar, U.S., 1997. Heuristics for the plate-cutting traveling salesman problem. *IIE Transactions*, 29 (9), 719–731.

Imahori, S., *et al.*, 2008. Generation of cutter paths for hard material in wire EDM. *Journal of Materials Processing Technology*, 206 (1-3), 453–461.

Jing, Y. and Zhige, C., 2013. An Optimized Algorithm of Numberical Cutting-Path Control in Garment Manufacturing. *Advanced Materials Research*, 796, 454–457.

Kim, Y., Gotoh, K., and Toyosada, M., 2004. Global cutting-path optimization considering the minimum heat effect with microgenetic algorithms. *Journal of Marine Science and Technology*, 9 (2), 70–79.

Kumar, D.V.S., 1998. Optimization of the nibbling operation on an NC turret punch press. *International Journal of Production Research*, 36 (7), 1901–1916.

Lee, M.K. and Kwon, K.B., 2006. Cutting path optimization in CNC cutting processes using a two-step genetic algorithm. *International Journal of Production Research*, 44 (24), 5307–5326.

Manber, U. and Israni, S., 1984. Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutling. *Journal of Manufacturing Systems*, 3 (1), 81–89.

Moreira, L., *et al.*, 2007. Heuristics for a dynamic rural postman problem. *Computers & Operations Research*, 34 (11), 3281–3294.

Noon, C. and Bean, J., 1991. An efficient transformation of the generalized traveling salesmen problem. *INFOR*, 31 (1), 1993.

Rao, Y., *et al.*, 2006. An integrated manufacturing information system for mass sheet metal cutting. *The International Journal of Advanced Manufacturing Technology*, 33 (5-6), 436–448.

Renaud, J. and Boctor, F.F., 1998. An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal Of Operational Research*, 108 (3), 571–584.

Rodrigues, A.M. and Soeiro, F.J., 2012. Cutting path as a Rural Postman Problem : solutions by Memetic Algorithms. *Journal of Combinatorial Optimization Problems and Informatics*, 3 (1), 22–37.

Rosenkrantz, D.J., Stearns, R.E., and Lewis, P.M., 1974. Approximate Algorithms for the Traveling Salesman Problem. *In*: *Fifteenth Annual IEEE Symposium on Switching and Automata Theory*, 33–42.

Sherali, H.D., Sarin, S., and Boothra, A., 2005. New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations Research Letters*, 33 (1), 62–70.

Srivastava, S.S., *et al.*, 1969. Generalized traveling salesman problem through n sets of nodes. *CORS*, 7, 97–101.

Verlinden, B., *et al.*, 2007. Sequencing and scheduling in the sheet metal shop. December,

In: E. Levner, ed. *Multiprocessor Scheduling: Theory and Applications*. Vienna: Itech Education and Publishing, chap. 20, p. 436.

Wang, G.G. and Xie, S.Q., 2005. Optimal process planning for a combined punch-and-laser cutting machine using ant colony optimization. *International Journal of Production Research*, 43 (11), 2195–2216.

Xie, S.Q., *et al.*, 2009. Optimal process planning for compound laser cutting and punch using Genetic Algorithms. *International Journal of Mechatronics and Manufacturing Systems*, 2 (1/2), 20–38.

Yang, W.B., *et al.*, 2010. An Effective Algorithm for Tool-Path Airtime Optimization during Leather Cutting. *Advanced Materials Research*, 102-104, 373–377.

Appendix A. Test instances

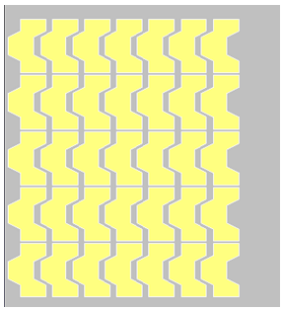

Figure A1. Instance 10

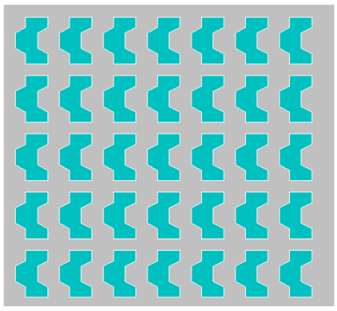

Figure A2. Instance 10b

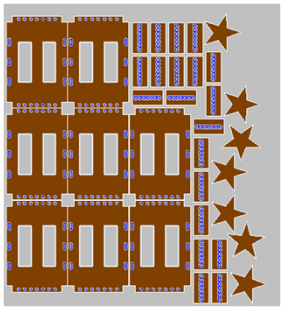

Figure A3. Instance 11

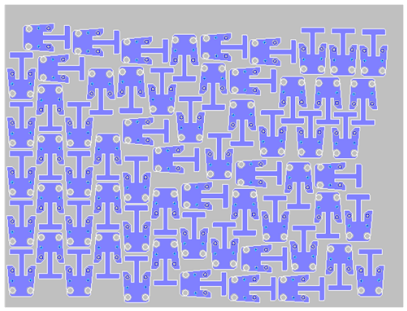

Figure A4. Instance 12

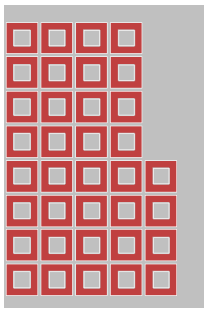

Figure A5. Instance 13

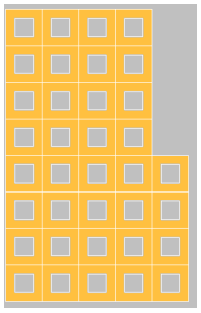

Figure A6. Instance 14

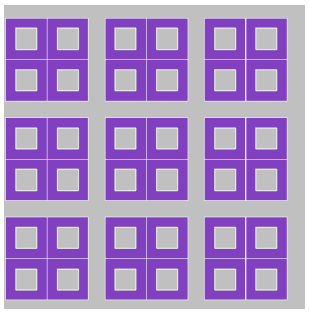

Figure A7. Instance 15

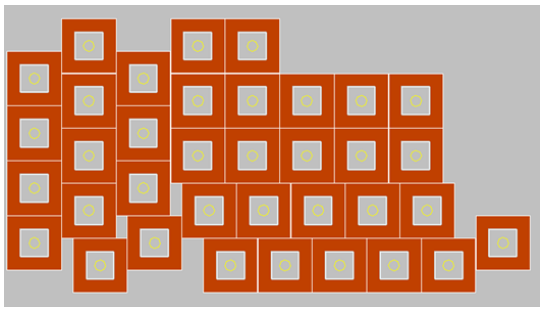

Figure A8. Instance 16

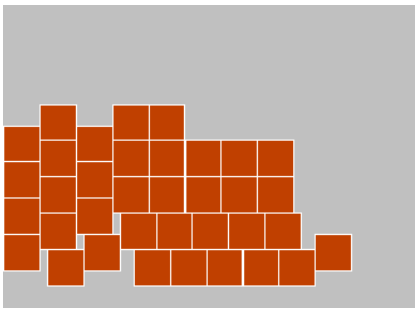

Figure A9. Instance 16b

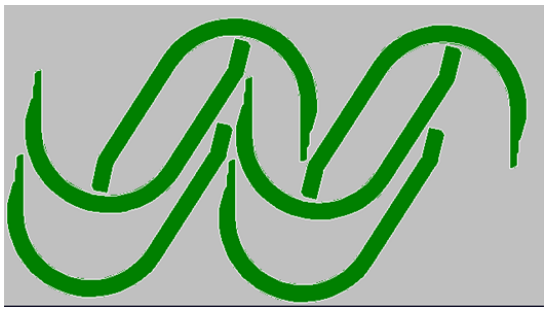

Figure A10. Instance 17

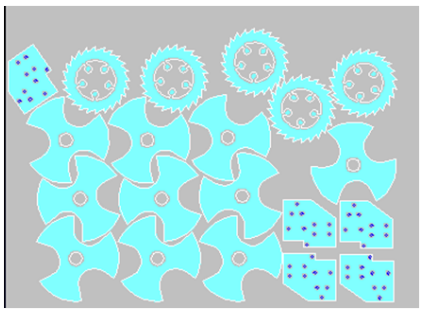

Figure A11. Instance 18

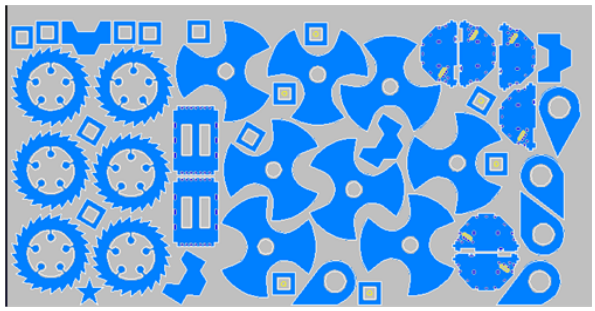

Figure A12. Instance 19

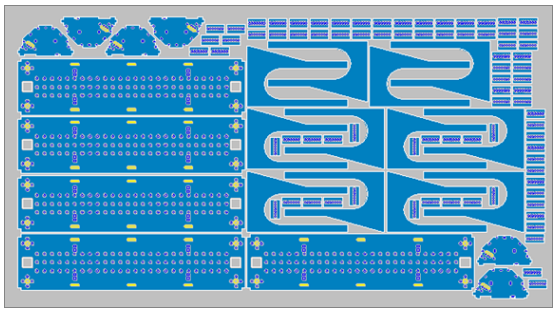

Figure A13. Instance 20

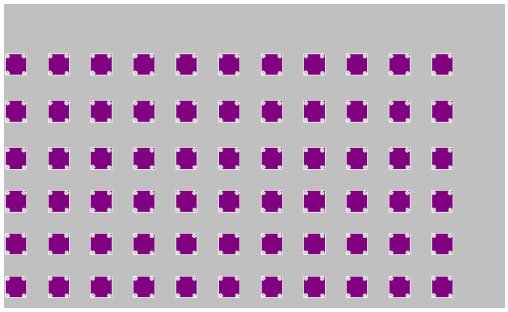

Figure A14. Instance 21

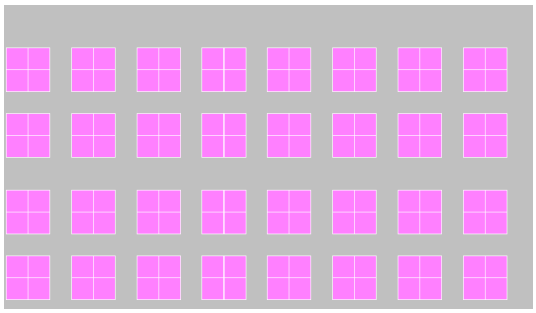

Figure A15. Instance 22

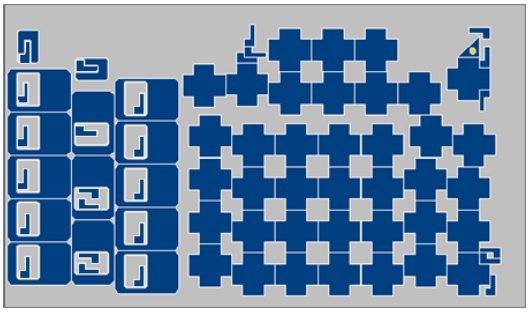

Figure A16. Instance 23

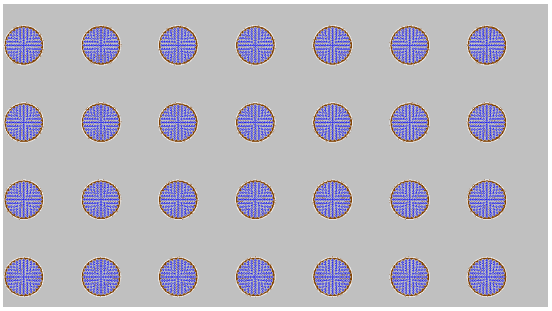

Figure A17. Instance 24

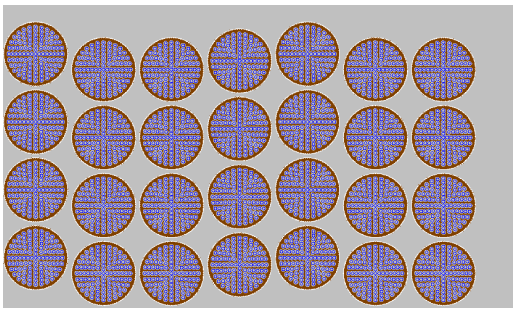

Figure A18. Instance 24b

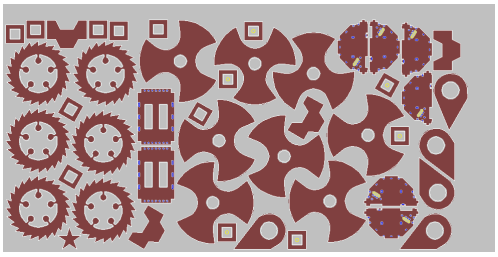

Figure A19. Instance 25

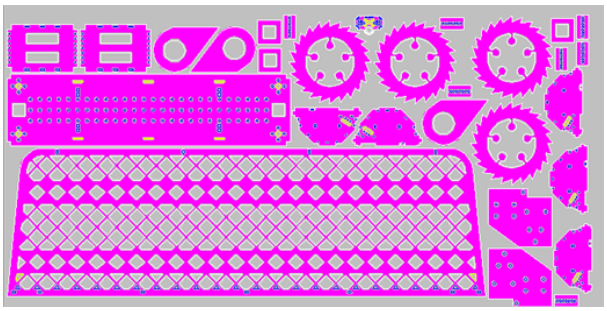

Figure A20. Instance 26

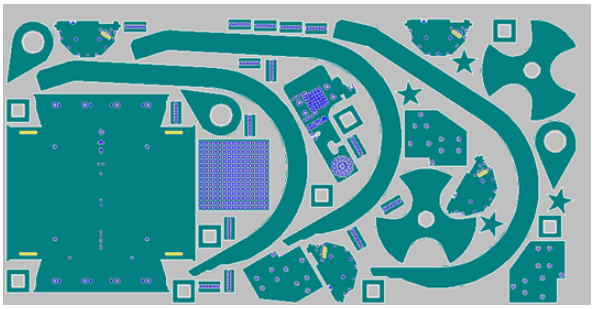

Figure A21. Instance 27

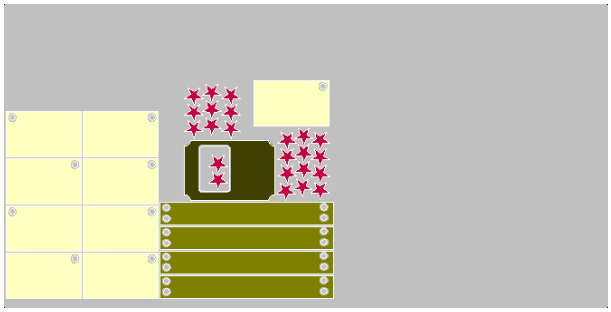

Figure A22. Instance 28

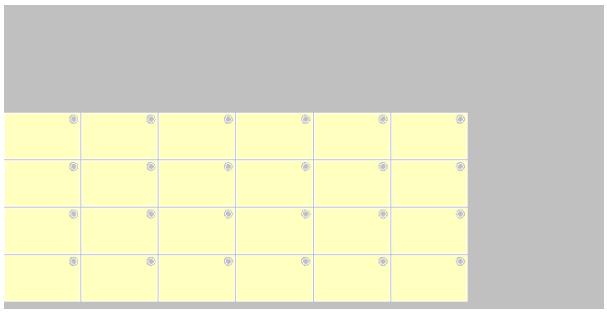

Figure A23. Instance 29

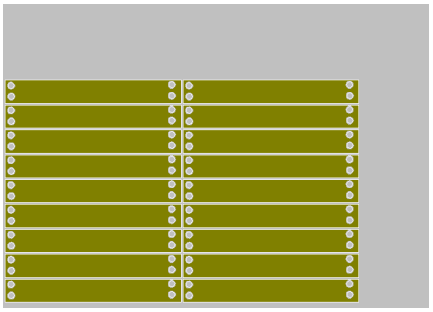

Figure A24. Instance 30

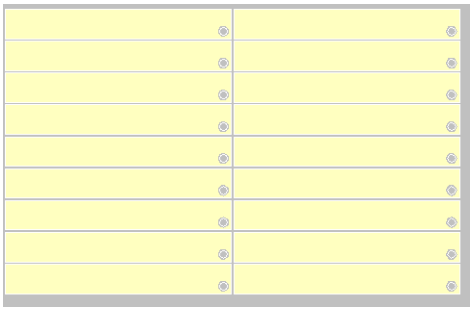

Figure A25. Instance 34

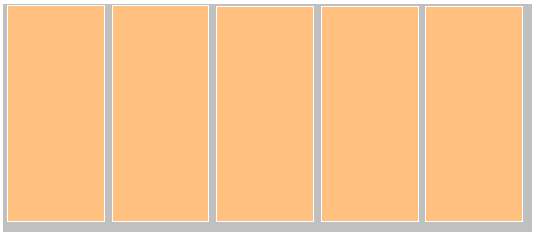

Figure A26. Instance 35

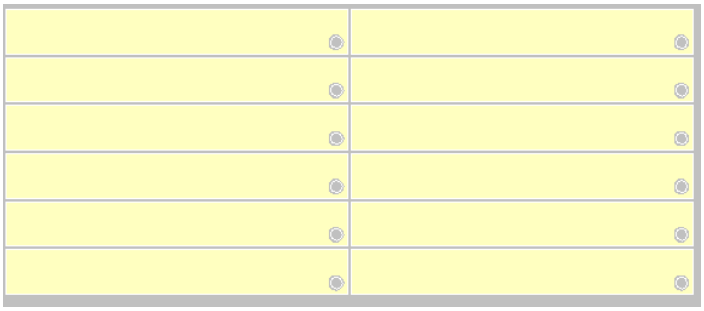

Figure A27. Instance 36

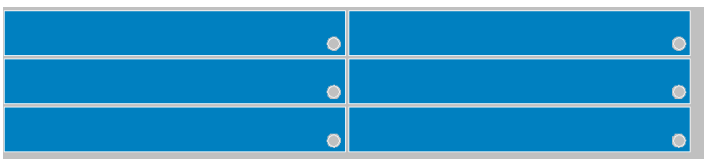

Figure A28. Instance 37

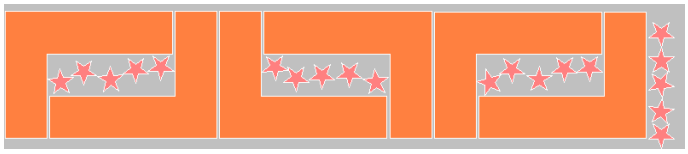

Figure A29. Instance 38